Lymphoma Data Augmentation

Mohammadhossein Akbari Moafi[†]

***Abstract*—Designing deep learning models for medical images has always been an appealing topic to research. But these models need large datasets during their training stage. On the other hand, medical images mostly are not accessible due to few number of patient or privacy reasons. In particular, to training deep learning models, lack of data will lead to overfitting and poor or in accurate result. Therefore, in this concept the rule of data augmentation will be crucial. In this work, to increase the number of images artificially, I present eight different approaches (four individual approaches and four combinations of these four) consist of Geometric Transformation and Feature Transform and Noise Injection where each of them enlarge the dataset size one to five times (Original Dataset Size + Generated Images by each APP). In this classification task Transfer learning with three pre-trained CNN models, AlexNet, ResNet50 and DenseNet-121 utilized to test and compare the result of proposed approaches on Lymphoma dataset with 374 image. I report 98.67 percent accuracy with APP1, APP3, APP5, APP6 and APP8 from DenseNet-121 and ResNet-50 models. The source code is available at https://github.com/r4stin/Lymphoma-Data-augmentation**

***Index Terms*—Deep Learning; Data Augmentation; CNN; Feature Transform; Transfer learning**

I. Introduction

Medical image interpretations are mostly performed by medical professionals like clinicians and radiologists. However, the variations among different experts and complexities of medical images make it very difficult for the experts to diagnose diseases accurately all the time. So the recommender systems rule will be significant in this field, where they help the experts to make objective, rapid and accurate diagnoses. But having reliable amount of data has always being a challenge. Training deep learning models, need large number of images. However, the number of medical images are always limited, therefore it will leads to overfitting or inaccurate result. Overfitting occurs when an algorithm fits too closely or even exactly to its training data, resulting in a model that can not make accurate predictions or conclusions from any data other than the training data.

To tackle this issue data augmentation plays a crucial rule. Data augmentation is the process of generating new data from the existing data. Data augmentation techniques divided into; Supervised, Label preserving, and Unsupervised.

Supervised data augmentation can be equivalently seen as constructing an augmented labeled set from the original supervised set and then training the model on the augmented set. GAN(Generative Adversarial Networks) [1] and its improved methods can be categorized into the supervised methods.

[†]Department of Information Engineering, University of Padova, email: {mohammadhossein.akbarimoafi}@studenti.unipd.it

Special thanks / Professor Loris Nanni.

Unsupervised data augmentation means the augmentation methods are not related to data labels [2]. For image classification tasks, some categroy-free image transformation methods are employed to generate new samples from the training set. Geometric transformation is a commonly-used unsupervised data augmentation.

There are cases where training data is expensive or difficult to collect. Therefore, there is a need to create high-performance learners trained with more easily obtained data from different domains. This methodology is referred to as transfer learning which is a useful way to reduce overfitting. In this work, I trained and tested my approaches by utilizing transfer learning on three pre-trained AlexNet, Resnet50, and DenseNet121 models on the Lymphoma dataset.

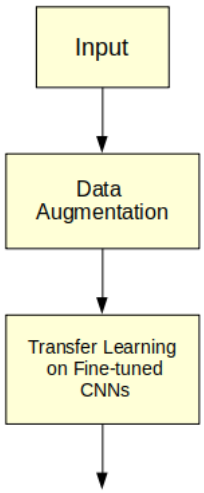


Fig. 1: Schematic of the proposed system workflow.

II. Related Work

Although machine learning and deep learning has resolved very complex medical issues, but lack of data especially in clinical task always raising issues. To tackle this problem, [3] is a good example in bioimage classification, where the researchers were able to detect breast cancer with a dataset containing 400 images of 4 classes by applying multiple augmentation techniques e.g 50 different color augmentation and using pre-trained ResNet-50, InceptionV3 and VGG-16 networks for feature extraction. Furthermore, [4], [5] where the researchers introduced multiple feature transformation approaches using Principle Component Analysis (PCA) and Discrete Cosine Transform (DCT).

III. Methodology

In this study, the original dataset comprises 374 images, with 299 allocated for the training set and the remaining 75 images reserved for the test set. Additionally, 20 percent of the training set is used as the validation dataset. During the training phase, I randomly extracted 224x224 images from the original 312x417 images. Additionally, by converting the images into PyTorch tensors, I normalized all the images to a range of 0 to 1. In the test phase, I did not apply any pre-processing; I only converted all the images into PyTorch tensors.
First approach only consist of geometrical transformation, the second approach is about noise injection and APP3, APP4 are based on two common feature transforms: PCA and DCT and the remaining approaches are the different combination of these approaches.

A. Data augmentation methods

Principal component analysis (PCA) is a technique for reducing the dimensionality but at the same time minimizing information loss. PCA by identifying the principal components of the image data, which capture the most variation in the dataset. By retaining only the most significant components and discarding the less significant ones, PCA achieves compression.
DCT is an another approach to image compression, which is more efficient in term of computational complexity compare to PCA. DCT transforms the image data into a set of coefficients representing different frequency components. These coefficients are then quantized and encoded to achieve compression. Unlike PCA, which operates on the covariance matrix of the data, DCT is not data-dependent, leading to reduced computational complexity.

App1: This approach consist of three common geometric transformation. The image is randomly reflected in both the left-right and the top-bottom directions with 50% probability [4] in addition to an Affine transform. In Affine transform, rotation between [-10, 10] interval and translation consists of shifting the image by (0, 0.05) tuple of maximum absolute fraction for horizontal and vertical translations, scaling factor interval [1, 1.2] is randomly sampled from the range $a \leq$ scale $\leq b$ finally randomly shearing in interval [0, 30] are adjusted as Affine transform parameters. This approach generates three new images for each image in original dataset.

This approach diversifies data by applying reflection, rotation (-10 to 10 degrees), and affine transformations (translation, scaling, shearing). Randomness (50% probability ensures uniqueness and improves the model's robustness to unseen data.

APP2: The second approach is noise injection [6]. This approach consists of two functions. The first function generates Gaussian noise with mean 0 and standard deviation 1, matching the shape of the input image. The noise is added to the image with a scaling factor of 0.2. The second function, first multiply the input image by 2. If the resulting pixel values are less than or equal to 1, noise is added using the same scaling factor as before. If the pixel values exceed 1, first subtract 1 from that pixel then add 1 to ensure result starts from 0, then Gaussian noise is applied to this excess, similar to the previous case. After applying noise, the result is negated and then added to 2 to revert the initial scaling applied to the image. To normalize it is subtracted from 2 to bring the pixel values back within the range [0, 1]. This approach generates two new images for each image in original dataset.

This approach help the model to learn a wider range of possible inputs, making it more robust to noise and less likely to overfit to the training data.

APP3: This is a data augmentation (DA) approach based on PCA, inspired by [5], [6] which apply PCA on each of 3 channels of input image with two Gaussian disturbance (0, 0.1) and (0, 0.2) with probability of 0.5 to generate the new image. Below you can see the pseudocode of this approach at *Algo* 1. This approach generates one new image for each image in original dataset.

This approach introduces Gaussian noise with two distrurbance, providing controlled randomness to the image data. This randomness simulates real-world variations, making the augmented image more representative of the real one. Also, PCA ensures that the most significant features and variations within each channel are retained. By focusing on the principal components, PCA captures the essential structure of the data. The perturbation based on eigenvalues and eigenvectors preserves this structure while introducing variability, ensuring that the augmented data remains realistic and meaningful.

APP4: The fourth approach apply DCT on each channels of the input image and computes and set standard deviation of each channel as a constant to scale a random perturbation application in range [-0.5, 0.5] [4] Perturbations are applied to selected DCT coefficients (ten percent of each channel), preserving specific coefficients like (1,1). Finally, Inverse DCT transformation reconstructs the image. You can see the pseudocode at *Algo* 2. This approach generates one new images for each image in original dataset.

By applying DCT, the image data is transformed into the frequency domain where specific frequency components can be targeted for perturbation. This allows for more controlled and meaningful alterations to the image compared to arbitrary pixel-level changes. Computing the standard deviation of the DCT coefficients and scaling the perturbations ensures that the modifications are proportionate to the image's frequency characteristics. This avoids dramatics modification on input image.

APP5: Fifth approach is combination of APP1 and APP3. This approach generates four new images for each image in original dataset.

APP6: Sixth approach is combination of APP2 and APP3. This approach generates three new images for each image in original dataset. This approach generates three new images for each image in original dataset.

APP7: Seventh approach is combination of APP3 and APP4. This approach generates two new images for each image in original dataset.

APP8: Eighth approach is combination of APP1, APP3 and APP4. This approach generates five new images for each image in original dataset.

The data augmentation methods described above are shown in *Fig.2*.

Algorithm 1 Pseudocode of APP3

Input: Image: 312x417x3
Output: Perturbed_Image: 224x224x3
channels ← [red, green, blue]
scale_value(with probability 0.5) ← [0.1, 0.2]
for channel **in** channels **do**
  # *Calculate Gaussian distribution*
  alpha ← $N(0, 1)$ * *scale_value*
  # *Apply perturbation on each channel*
  perturbation ← $MatrixMultiplication$(eigenvectors.T, alpha * eigenvalues)
end for
# *Reshape 1D Image into 3 Channels Image*
perturbed_image ← $Reshape(channels)$
Apply Crop and Resize
perturbed_image ← $Resize(perturbed\_image)$
perturbed_image ← $RandomCrop(perturbed\_image)$
return perturbed_image

IV. Setup and Models

A. Setup

For this experiment, the Kaggle platform was utilized, employing the following primary hardware and software configurations:

- **CPU:** 4 core and 30GB RAM.
- **GPU**: Nvidia Tesla P100, 4 cores and 29GB RAM
- **Development language:** python 3.10.13.
- **Deep learning development platform:** Pytorch 2.1.2.

B. Models

During training, I utilized the CrossEntropy loss function to handle this multi label classification task, Adam optimizer for optimization, and a learning rate of 0.01 initially, which

Algorithm 2 Pseudocode of APP4

Input: Image: 312x417x3
Output: NewImage: 224x224x3
(y, x, z) ← *Shape of Image*
for channel **in** range(z) **do**
  # *Apply DCT*
  DCTImage ← $DCT(channel)$
  # *Apply on* 10% *of pixels*
  Indices ([False, True] ← $[0.9, 0.1]$
  Sigma ← $std(DCTImage[Indices]$
  # *Random perturbation in range* $[-0.5, 0.5]$
  Random_p ← $U(-0.5, 0.5) * sigma$
  DCTImage[Indices] ← $DCTim[Indices]+Random\_p$
  # $Except\ for\ DCTImage(1, 1)$
  # *Apply Inverse DCT*
  Image ← $IDCT(DCTImage)$
end for
# *Apply Crop and Resize*
NewImage ← $Resize(Image)$
NewImage ← $RandomCrop(Image)$
return $NewImage$

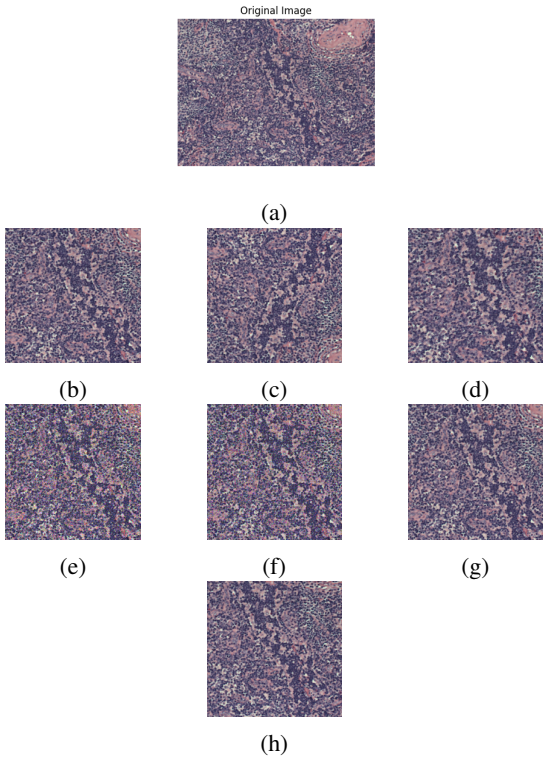


Fig. 2: a) Original Image b) Random Horizontal Flip c) Random Vertical Flip d) Random Affine e) Noise Injection (method 1) f) Noise Injection (method 2) g) PCA perturbation h) DCT perturbation

caused overshooting, and later reduced it to 0.0001 which led to good convergence. After testing on 16, 32 and 64 batch sizes in this report only the result of batch size 32 reported.
Output Layers: The models all have a fully connected layer outputting 1000 classes. To employ transfer learning, I fine-tuned all the models by modifying the last fully connected layer (FC) of each model according to our classification task.

Early Stopping: A custom function defined for early stopping which monitors a specific metric (validation loss) during training and stops the training early if the metric not decreasing for a number of epochs. This function will activate when the epoch number is greater than 20, this threshold will help us to reach a good convergence. To select the number 20, all models were trained for 200 epochs to determine the optimal threshold. It also keeps track of the best model weights and stops training if the monitored metric does not meet the the metric lower than the best one for a given maximum number of epochs (10 consecutive epochs).

- **AlexNet:** AlexNet [7] consists of eight layers: five convolutional layers followed by three fully connected layers. It employs techniques such as ReLU activation functions, overlapping pooling, and dropout regularization. The network architecture was pivotal in demonstrating the effectiveness of deep learning in image classification tasks.

  Base Model: First model is based on a "AlexNet" architecture where you can see at *Fig.3*.
  Fully Connected Layers: The final two classifiers, initially configured with input dimensions of 4096 and output dimensions of 4096, and input dimensions of 4096 with an output of 1000, have been modified to now have input dimensions of 4096 and output dimensions of 1024, and input dimensions of 1024 with an output of 3.

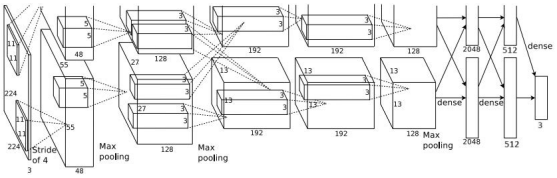


Fig. 3: AlexNet Fine-tuned Architecture.

- **Resnet-50:** ResNet-50 [8] is a deep convolutional neural network architecture that belongs to the ResNet (Residual Network) family. ResNet-50 specifically refers to a variant of ResNet that consists of 50 layers, including convolutional layers, pooling layers, and fully connected layers. ResNet-50 is known for its innovative use of residual connections, which allow information to bypass certain layers and propagate directly to subsequent layers. This helps mitigate the degradation problem often encountered in deep neural networks, where adding more layers can lead to diminishing performance. ResNet-50 has achieved impressive results in various computer vision tasks, including image classification and object detection, by effectively capturing hierarchical features at different scales.

  Base Model: The second model is based on a "ResNet-50" architecture where you can see at *Fig.4*.
  Fully Connected Layers: The FC of this model, initially configured with input dimensions of 2048 and output dimensions of 1000, have been modified to now have input dimensions of 2048 and output dimensions of 3.

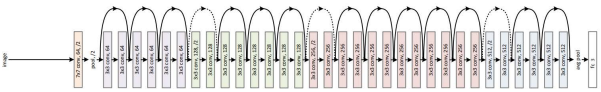


Fig. 4: ResNet-50 Fine-tuned Architecture.

- **DenseNet-121:** DenseNet-121 [9] comprises 121 layers, featuring a densely connected network structure where each layer receives direct input from all preceding layers. This connectivity pattern fosters feature reuse and enhances gradient flow throughout the network, leading to improved parameter efficiency and feature propagation. The architecture consists of four dense blocks, each followed by a transition layer to control the growth of feature maps and reduce dimensionality. DenseNet-121 employs bottleneck layers, batch normalization, and ReLU activation functions, contributing to its effectiveness in learning discriminative features from images.

  Base Model: The third model is based on a "DenseNet-121" architecture where you can see at *Fig.5*.
  Fully Connected Layers: The last classifier of this model, initially configured with input dimensions of 1024 and output dimensions of 1000, have been modified to now have input dimensions of 1024 and output dimensions of 3.

V. Results

In *Tab.1* and *Tab.2*, respectively, I presented the number of training size and the accuracy results of various augmentation strategies. During training, most approaches achieved a validation accuracy of 100%. As anticipated due to the architecture and complexity of the DenseNet-121 model, this model achieved the best results. In *Fig.6*, I have reported the best performance of DenseNet-121 in terms of training and validation loss over epochs.

The AlexNet Model achieved average of 92.00 % accuracy which was the lowest during testing. The model's performance

TABLE 1: Size of Training Dataset for each of approaches.

Approaches	Original	APP1	APP2	APP3	APP4	APP5	APP6	APP7	APP8
Dataset size	299	1196	897	598	589	1495	1196	897	1794

TABLE 2: Different Augmentation approaches in term on Accuracy on Lympgoma dataset

Models	APP1	APP2	APP3	APP4	APP5	APP6	APP7	APP8	Avg
AlexNet	93.33	94.67	89.33	89.33	97.33	89.33	92.00	94.67	92.83
ResNet-50	**98.67**	96.00	**98.67**	**97.33**	96.00	97.33	96.00	94.67	96.83
DenseNet-121	**98.67**	**97.33**	97.33	**97.33**	**98.67**	**98.67**	**97.33**	**98.67**	**98.00**
Avg	96.89	96.00	95.11	94.66	**97.33**	95.11	95.11	96.00	

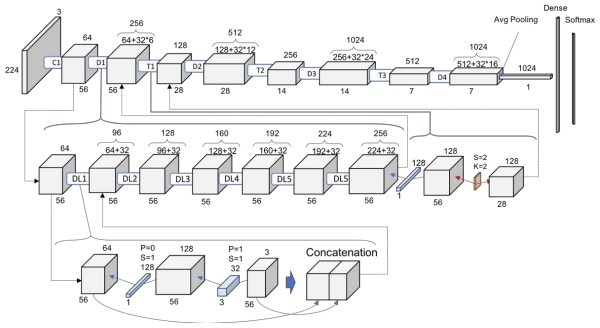
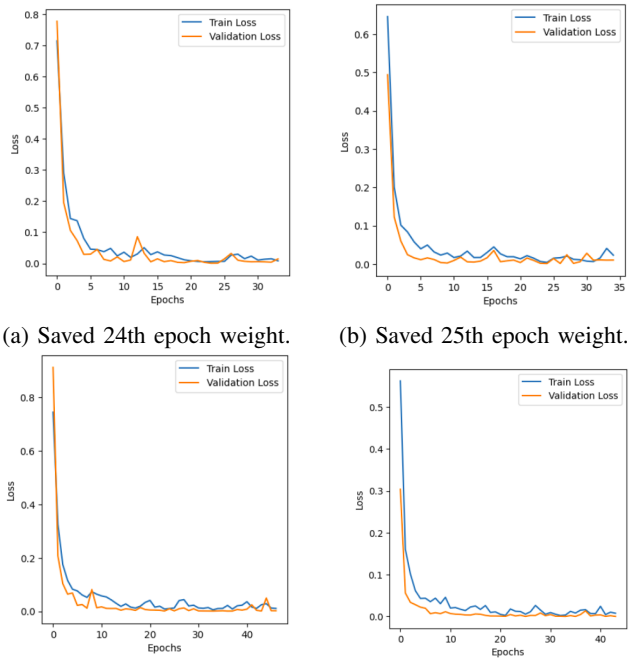


Fig. 5: DenseNet-121 Fine-tuned Architecture.



(a) Saved 24th epoch weight. (b) Saved 25th epoch weight.

(c) Saved 37th epoch weight. (d) Saved 34th epoch weight.

Fig. 6: DenseNet-121 best performance. a) APP1 b) APP5 c) APP6 d) APP8

on the best approach was on APP5 with 97.33% accuracy and it's overall performance over all the approaches was 92.83%.

ResNet50, a deeper network architecture, consists of multiple residual blocks, enabling the network to effectively capture complex patterns and features in the input images. The model demonstrated improved performance compared to the AlexNet network. The best performance in term of accuracy achieved by ResNet50 was 98.67% on APP1, APP3 and average accuracy of 96.83% over all the approaches.

DenseNet-121, with a more complex network structure, resulted in enhanced prediction accuracy. The best performance in term of accuracy achieved by DenseNet-121 was 98.67% which is repeated with APP1, APP5, APP6, APP8 and average accuracy of 98.00% over all approaches showcasing a substantial improvement in the average deviation compared to both the AlexNet and ResNet-50.

The augmentation techniques (geometric transformations, PCA, noise injection) each enhance different aspects of the data, helping models learn more robust and generalizable features. The complex architectures of ResNet50 and DenseNet-121 particularly benefit from these preprocessing methods due to their ability to capture and leverage intricate patterns and relationships within the data.

Overall, the experimental results demonstrate that deeper and more complex models, such as ResNet50 and DenseNet-121, exhibit improved accuracy in Lymphoma prediction task

compared to the AlexNet network. In *Tab.3* the classification report of DenseNet-121 at its best performance is shown.

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	29
1	1.00	0.96	0.98	27
2	0.95	1.00	0.97	19
Macro Avg	0.98	0.99	0.99	75
Weighted Avg	0.99	0.99	0.99	75

TABLE 3: Classification Report of best result on DenseNet-121

VI. Concluding Remarks

My experiments showed that CNN's clearly profit by using augmentation techniques (Feature Transform) additional to the basic augmentation techniques (cropping and flipping).

Future work in this area could explore the the supervised technique, such as GANs, to further improve malignancy detection accuracy. Additionally, incorporating larger and more diverse datasets, along with data augmentation techniques, also will help to enhance the robustness and generalization capabilities of the models.

In conclusion, my study demonstrates the effectiveness of augmentation techniques. These approaches have the potential to assist healthcare professionals in making accurate and efficient assessments, contributing to improved clinical decision-making and patient care.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[2] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[3] A. Rakhlin, A. Shvets, V. Iglovikov, and A. A. Kalinin, "Deep convolutional neural networks for breast cancer histology image analysis," *CoRR*, vol. abs/1802.00752, 2018.

[4] L. Nanni, S. Brahnam, S. Ghidoni, and G. Maguolo, "General purpose (genp) bioimage ensemble of handcrafted and learned features with data augmentation," *ArXiv*, vol. abs/1904.08084, 2019.

[5] L. Nanni, M. Paci, S. Brahnam, and A. Lumini, "Feature transforms for image data augmentation," *CoRR*, vol. abs/2201.09700, 2022.

[6] S. Jia, W. Ping, J. Peiyi, and S. Hu, "Research on data augmentation for image classification based on convolution neural networks," *2017 Chinese Automation Congress (CAC)*, pp. 4165–4170, 2017.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[9] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016.